Universidad
Carlos III de Madrid
www.uc3m.es

TRABAJO FIN DE GRADO

Título: Diseño e implementación de un reconocedor de habla híbrido en Matlab

Autor: Miguel Ángel Mota Martín

Titulación: Grado en Ingeniería de Sistemas Audiovisuales

Profesor: Dra. Ascensión Gallardo Antolín y Dra. Carmen Peláez Moreno

Fecha: Junio 2013

Agradecimientos

Durante todo el tiempo que ha durado esta etapa de mi vida, muchas personas se han cruzado en mi camino. Unas se han quedado más tiempo, otras menos. Algunas, de hecho, todavía siguen a mi lado. Hayan estado mucho o poco tiempo, cada una ha aportado algo nuevo a mi vida. Por ello, gracias a todas ellas.

Y por supuesto, gracias a las personas que no se han cruzado, sino que siempre han estado presentes y ofreciendo un apoyo incondicional.

Resumen

En este trabajo hemos abordado la implementación de las primeras etapas de un reconocedor automático de habla en Matlab con una arquitectura híbrida ANN/HMM (Artificial Neural Networks / Hidden Markov Models). En particular se han implementado los módulos de adquisición, extracción de características y parte del modelado acústico.

Para la adquisición, se han utilizado librerías estándar de matlab para poder realizar la lectura de la base de datos ISOLET. Esta librería es ampliamente conocida en el área del reconocimiento automático de habla.

Posteriormente, se ha utilizado la librería Voicebox para obtener los coeficientes MFCC (Mel Frequency Cepstral Coefficients) así como los coeficientes dinámicos correspondientes. Además, se ha añadido un procedimiento para construir un contexto para cada uno de los vectores de parámetros.

Por último, se ha realizado una búsqueda y posterior selección de una librería matlab para la implementación de MLP (Multi-Layer Perceptrons) con los requisitos necesarios para su posterior integración con los HMM. Así finalmente, se ha implementado un módulo de estimación de las probabilidades a posteriori de los vectores anteriormente descritos dada las 28 posibles clases de fonemas de nuestro entorno de experimentación.

Abstract

In this project, we have implemented the first steps of an Automatic Speech Recognizer (ASR) in Matlab employing a hybrid ANN/HMM (Artificial Neural Networks / Hidden Markov Models) scheme. In particular, modules for the acquisition, feature extraction and acoustic modeling (partly) have been implemented.

Standard Matlab libraries have been employed for the acquisition module enabling the sequential reading of the well known ISOLET database.

Then, Voicebox, a library specifically designed for speech processing, has been employed for the computation of MFCC (Mel Frequency Cepstral Coefficients). Besides, a procedure for the construction of an acoustical context for each of the feature vectors has been included.

Finally, a search process and subsequent selection of a neural network matlab library for the implementation of MLP (Multi-Layer Perceptrons) with the requirements needed for the foreseen integration into HMM has been carried out. Lastly, the 'a posteriori' probabilities estimation module for each of the feature vectors previously described given the 28 possible phonetic labels of our experimental testbed was implemented.

Universidad Carlos III de Madrid
www.uc3m.es

ÍNDICE

1 INTRODUCCIÓN

1.1. MOTIVACIÓN

Desde que existen máquinas tal y como las concebimos hoy en día, es necesario que exista una manera de transmitir órdenes en sentido hombre-máquina.

Esta comunicación va desde los botones para activar ciertos mecanismos, hasta programación software que indica a la máquina una serie de acciones a seguir según unas condiciones.

Últimamente, cada vez cobra más importancia la comunicación por voz, por su sencillez y naturalidad, y se busca que las máquinas sean capaces de entender esas órdenes.

Debido a esto, es necesario la investigación en el reconocimiento del habla para que dichas máquinas sean capaces de entender la orden dada y ejecutar las acciones posteriores correspondientes.

1.2. OBJETIVOS

El objetivo de este trabajo es construir un sistema de reconocimiento automático del habla basado en redes neuronales en Matlab

Este Trabajo Fin de Grado se forma de varios módulos que trabajan de manera secuencial y que unidos, forman el sistema total.

El primero de los módulos lo forma la base de datos de archivos de audio. Se utilizará una base cuyo uso está muy extendido en cualquier estudio relacionado con la extracción de coeficientes acústicos.

El siguiente módulo consiste en la extracción de parámetros acústicos de estas muestras almacenadas en la base de datos, lo que nos permitirá la formación de distintos conjuntos para su posterior uso. En este caso, se utilizarán unos parámetros mel-cepstrales, los MFCC ("Mel Frequency Cepstrum Coefficients").

Y por último, está el módulo que forma el reconocedor propiamente dicho. Este bloque se forma de diferentes partes. La parte más compleja es la red neuronal constituida para la clasificación de las muestras. Esta red neuronal recibe como entrada un conjunto de datos, que no son otra cosa que los coeficientes acústicos de una parte de la base de datos. Junto a este conjunto, recibe las etiquetas de cada elemento. Con el conjunto y las etiquetas, se entrena la red neuronal, de manera que ésta vaya ajustando sus pesos, para que genere una salida lo más parecida posible a las etiquetas de entrada. La segunda de las partes de este último bloque, la forma el clasificador. Este clasificador se basa en la salida generada por la red neuronal para, a cada entrada, asignarle una etiqueta de salida.

1.3. GUÍA DE LECTURA

Este documento está estructurado de una manera similar a cómo ha sido el desarrollo de este proyecto.

La primera de las partes nos habla de lo que trata el proyecto, el motivo por el cual se investiga en este campo y los objetivos a conseguir con este proyecto. Esta parte del documento, se puede asemejar a las primeras reuniones con las tutoras, las que sirvieron para fijar el proyecto

La segunda parte es la parte más teórica. Esta parte la constituyen los puntos 2 y 3 del documento. En ella, se recoge toda la información necesaria, y se toman conocimientos sobre conceptos y técnicas pocos conocidas en ese momento, y que se van a aplicar en el trabajo.

La siguiente parte es la parte más práctica. Esta parte la forma el punto 4 de este documento. Aunque, esta parte práctica esté toda contenida en un solo punto, éste se ha formado por dos tareas diferenciadas en el tiempo. Primero, se llevó a cabo la tarea descrita en el punto 4.1, la extracción de parámetros. Y una vez terminada esa tarea, se pasó a la tarea del punto 4.2., la creación y entrenamiento del reconocedor.

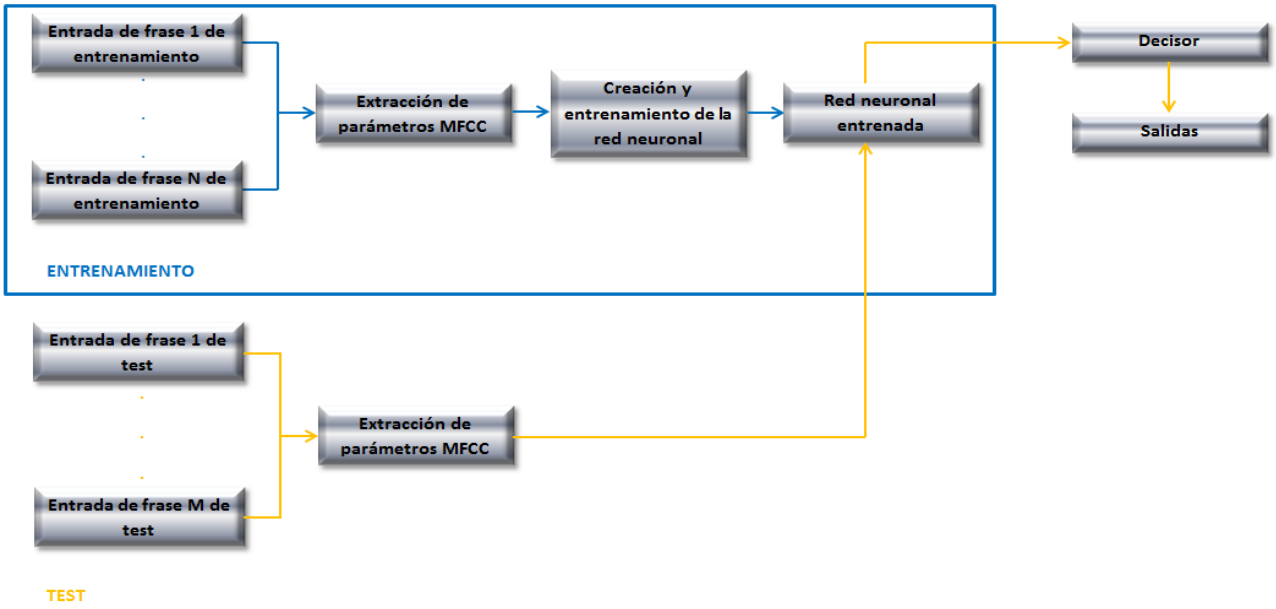
Ésta última tarea, la del punto 4.2. está muy ligada con el punto de experimentos (5.1), ya que los éstos se han obtenido modificando la configuración del reconocedor una y otra vez.

Por último, está el trabajo no realizado en este proyecto, las líneas futuras. Lo cual, podría ser trabajo que sirva de continuación de lo desarrollado en este proyecto.

2 FUNDAMENTOS DEL RECONOCIMIENTO DEL HABLA

2.1. ESTRUCTURA DE UN RECONOCEDOR DEL HABLA

El diagrama de bloques del reconocedor del habla construído es el siguiente:



El proceso total que lleva a cabo el reconocedor se puede separar en dos subprocesos.

El primero de ellos es el proceso de entrenamiento. Éste a su vez está compuesto por varios subprocesos. Lo primero que se hace es recoger la señal de audio y digitalizarla. Se graba a diferentes locutores, diciendo diferentes frases. Una vez que se tienen las señales de audio digitalizadas, se extraen los parámetros MFCC de cada una de ellas. Estos parámetros lo que hacen es dividir cada frase en un número de ventanas, y de cada una de ellas, extrae dichos parámetros. Además, en la etapa de entrenamiento de la red neuronal, cada conjunto de parámetros va etiquetado con el número de fonema al que pertenece. Se crea una red neuronal, definiendo las diferentes características como la función de entrenamiento, o la función de error.

El conjunto obtenido anteriormente, parámetros y etiquetas se introducen en la red neuronal para llevar a cabo el entrenamiento. La red, mediante una serie de iteraciones, también definidas, va ajustando sus pesos en función de las valores de las entradas y de las etiquetas.

De esta manera al término de las iteraciones, los pesos de la red quedan fijados, y por lo tanto, la red queda configurada.

El segundo paso que se lleva a cabo, el que se hace después del entrenamiento de la red, es el proceso de test. Este paso se compone también de otros bloques. El primero de ellos es la adquisición de señales de audio digitalizadas de diferentes locutores, y de diferentes frases. Una vez que se tienen estas señales, se extraen de ellas los parámetros MFCC de la misma manera que se hace en la fase de entrenamiento de la red. La diferencia es que el conjunto que se manda a la red neuronal, ya configurada, sólo está formado por los parámetros MFCC de las señales de audio, en esta ocasión, estos datos no van etiquetados. La red entrenada, tras recibir estas entradas, genera una salida con valores entre 0 y 1. El decisor aplica a cada entrada un fonema de salida, basándose en el mayor de los números asignados. De esta manera, a cada entrada de test, genera una salida.

Estos dos procesos descritos anteriormente, son secuenciales. Es decir, primero se tiene que llevar a cabo el proceso de entrenamiento, y después el proceso de test.

Los conjuntos de datos usados para entrenamiento y para test, por lo general, son diferentes. El conjunto de entrenamiento suele tener mayor extensión que el de test.

2.2. EXTRACCIÓN DE CARACTERÍSTICAS. PARÁMETROS MFCC

Los Mel Frequency Cepstral Coefficients son parámetros de la representación de la voz basados en la manera de escuchar humana **[6]**.

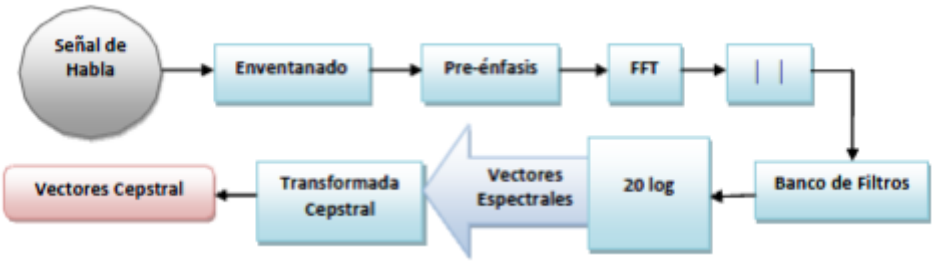
Estos parámetros son un tipo de coeficientes cepstrales derivados de la aplicación del Cepstrum.sobre una ventana de tiempo de la señal de voz. Por lo tanto, es necesario conocer de qué se trata el Cepstrum. El Cepstrum se define matemáticamente de la siguiente manera **[7]**:

$$Cepstrum(s[n]) = \hat{s}[n] = F^{-1}[log(|F[s[n]|)] = \frac{1}{2\pi}\int_{-\pi}^{\pi} log\left(\left|S(e^{j\omega})\right|\right)d\omega$$

En la fórmula anterior s[n] representa la convolución entre las dos componentes de información de la señal de voz, la excitación y el tracto vocal. Sin embargo, muy pocas veces se hace uso del Cepstrum de manera directa, debido a su alta vulnerabilidad a los efectos de canal.

Surgen de aquí, por tanto, los MFCC, para adaptarse a la manera de escuchar del oído humano. Estos parámetros son considerados una de las técnicas más importantes y más utilizadas para la parametrización de la voz. Estos parámetros consiguen una representación compacta y robusta de la señal de audio, para su posterior procesado.

Los pasos que se siguen para obtener dichos parámetros de una señal de voz vienen representados en el siguiente diagrama de bloques:



El **enventanado** se lleva a cabo porque la señal de voz es un proceso aleatorio no estacionario. Pero si se toman trozos pequeños de la señal, del orden de milisegundos, en esos tramos, la señal se puede considerar casi-estacionaria. El proceso de enventanado consiste en la segmentación de la señal en tramas consecutivas. Normalmente el tamaño de las ventanas es de 20 ms, y éstan van solapadas unas con otras para no perder información.

Generalmente el solapamiento entre ventanas se hace cada 10 ms, por lo que se obtienen parámetros MFCC cada 10 ms.

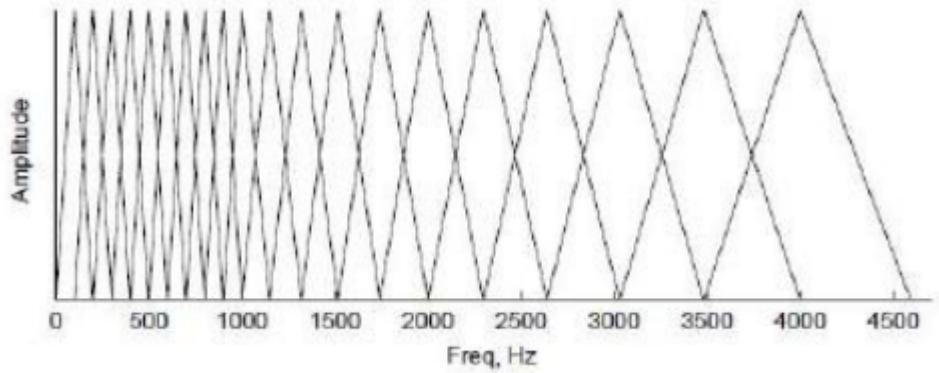
El **filtro de pre-énfasis** se introduce para compensar la atenuación de alrededor de 20dB/década que se produce en la señal de voz humana. Este paso no siempre se efectúa.

El siguiente paso consiste en la **Transformada de Fourier Discreta** de tamaño N de la señal enventanada. Se calcula dicha DCT de la siguiente manera:

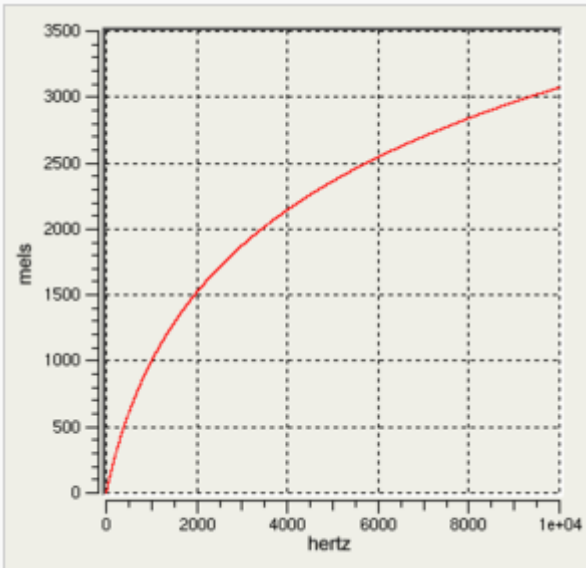
$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk}, 0 \leq k \leq N$$

El siguiente bloque lo que hace es descartar la fase de X[k], y quedarse sólo con el módulo de ésta. Es decir, desde este paso, se trata con la envolvente de la señal de voz.

El paso siguiente es el que diferencia a los MFCC del Cepstrum. La señal |X[k]| se pasa por un banco de filtros triangulares de área unidad. La separación entre los triángulos la marca la escala de frecuencias MEL. Un ejemplo de este tipo de banco de filtros es el siguiente:



Como se ha dicho anteriormente, estos filtros siguen la escala de frecuencias MEL. La relación entre Hz y mels, viene representada en la siguiente gráfica:



Para pasar de manera analítica de mels a Hz, o viceversa, existen unas relaciones:

$$m = 1127{,}01048 \log_e(1 + f/700) \qquad f = 700(e^{m/1127{,}01048} - 1)$$

Tras pasar la señal por el banco de filtros, es necesario calcular la energía en cada uno de los filtros H_m. La energía en cada filtro viene dada por la expresión:

$$E_m = \sum_{k=0}^{N-1} |X[k]|^2 H_m[k] \qquad 1 \leq m \leq F.$$

Después, se calcula el 20·log de estas energías calculadas. De esta manera, se pasa al dominio espectral logarítmico, y esto conlleva que los espectros de los filtros en bandas adyacentes presentan una alta correlación entre ellos. Por lo que los coeficientes espectrales originados guardan mucha relación estadística entre ellos.

Para deshacer esta correlación estadística, se lleva a cabo la DCT de estos coeficientes, llevándolos al dominio de la quefrencia y convirtiéndolos en coeficientes MFCC. Esto se consigue efectuando la siguiente operación:

$$c_{MFCC}[m] = \sum_{k=0}^{N-1} log(E_k)\, cos\left(m\left(k-\frac{1}{2}\right)\frac{\pi}{N}\right) \quad m = 1, \dots, F$$

Los parámetros MFCC obtenidos representan la envolvente espectral de la señal de voz captada, lo que representa una gran información sobre dicha señal.

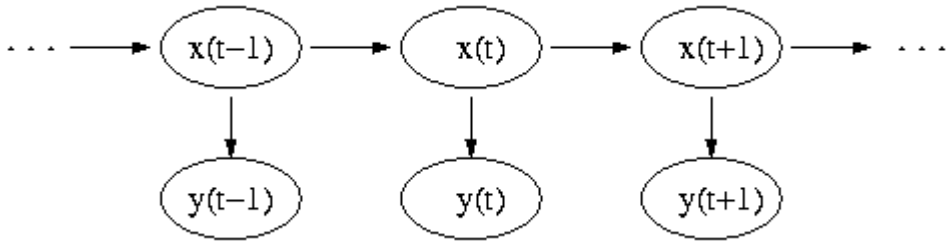
Si se quiere obtener más información de la señal, se pueden calcular, además de los MFCC otros parámetros conocidos como MFCC-Delta y los MFCC-Delta-Delta. Estos otros parámetros representan la evolución temporal de los fonemas en su transición a otros fonemas.

2.3. MODELADO ACÚSTICO

2.3.1 MODELOS OCULTOS DE MARKOV [8]

Un modelo oculto de Markov es un modelo estadístico. El objetivo de estos modelos es determinar los parámetros ocultos a partir de los parámetros observados. En un modelo oculto de Markov, el estado no es visible directamente, sino que sólo lo son las variables influidas por el estado. Cada estado tiene una distribución de probabilidad sobre los posibles símbolos de salida.

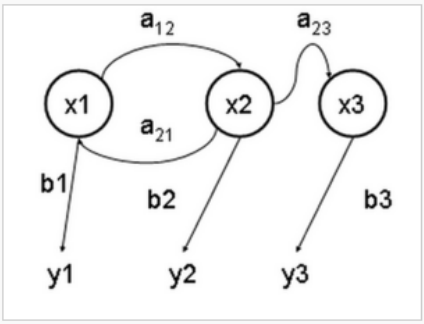
La arquitectura de un modelo viene representada en la siguiente figura:



En esta figura, cada óvalo representa una variable aleatoria, que puede moverse entre diferentes valores. Cuando se habla de variable aleatoria x, se está hablando de una variable aletoria oculta, y cuando se habla de variable aleatoria y, se está hablando de una variable aleatoria observada. La referencia de la variable t es la referencia temporal. Las flechas indican dependencia entre las diferentes variables aleatorias.

De este esquema se puede entender que el valor de la variable aleatoria x en el instante (t), sólo depende del valor de la variable aleatoria x en el instante (t-1). De la misma forma, el valor de la variable aleatoria y en el instante (t), sólo depende del valor de la variable aleatoria x en el mismo instante de tiempo (t).

De esta manera, se definen dos tipos de probabilidades. Las que relacionan un estado oculto con otro estado oculto (probabilidades de transición a_{ij}), y las que relacionan un estado oculto con un estado observado (probabilidades de salida b_i).



2.3.2 REDES NEURONALES

Éste es el tipo de modelado acústico utilizado en este trabajo. El próximo capítulo está dedicado en exclusividad a la explicación de este tipo de técnica de modelado acústico.

3 REDES NEURONALES ARTIFICALES

3.1. CONCEPTO [9] [10] [11] [12]

Las redes neuronales artificiales son una técnica de aprendizaje y procesado automático basada en las estructuras neurobiológicas encontradas en la naturaleza y en sus conexiones. Están formadas por elementos llamados neuronas interconectados entre sí.

Cada conexión se hace a través de unos pesos. Al recibir las entradas, estos pesos van siendo ajustados para conseguir la salida deseada. Para ello, las redes usan tres funciones:

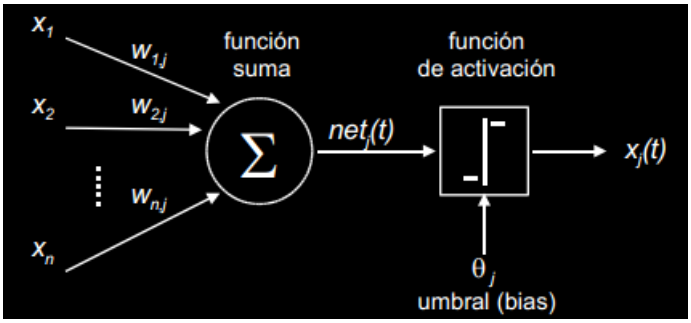
- **Función de propagación**: típicamente es una combinación lineal de la entrada con los pesos.

$$u_i(w,x) = \sum_{j=1}^{n} w_{ij} x_j$$

- **Función de activación:** la salida de la función de propagación se modifica mediante una función no lineal como puede ser la función sigmoidal o la función gaussiana.

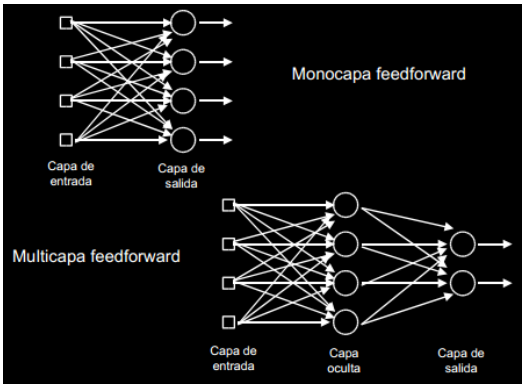
- **Función de transferencia:** se aplica a la salida de la función de activación para conseguir que todos los valores estén dentro de un intervalo (habitualmente [0,1]).

Por lo tanto, cada neurona se puede representar de la siguiente manera:

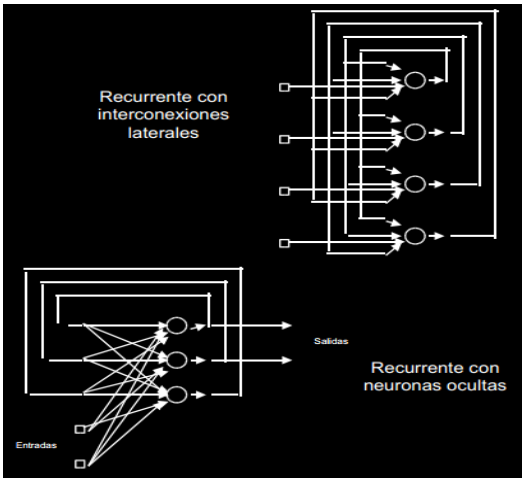


Se puede hacer una clasificación de las redes neuronales atendiendo al patrón de conexiones existentes en ella:

- En primer lugar, se pueden distinguir las redes en las que las señales van desde la capa de entrada hasta la capa de salida, sin llevar a cabo ningún tipo de ciclo: *Feedforward net*. Dentro de esta clasificación, se dividen también dependiendo del número de capas que tenga la red en *Redes Monocapa* como pueden ser el *perceptrón* o la *red Adaline*, o *Redes Multicapa* como puede ser el *perceptrón multocapa*.



- Por otro lado están las redes recurrentes, en las que existe al menos un ciclo cerrado de activación neuronal. Algún ejemplo son la *red Elman*, la *red Hopfield* o la *máquina de Boltzmann*.



Otra posible clasificación de las redes neuronales, es atendiendo a su forma de aprendizaje. De esta manera, existen diferentes tipos de aprendizaje:

- **Aprendizaje supervisado:** en este tipo de aprendizaje, a la red neuronal se le pasa la entrada con la salida deseada. La red va ajustando los pesos para que la salida dada se ajuste a la salida deseada.Éste es el tipo de aprendizaje que siguen redes como el *perceptrón* o la *red Adaline*. Se puede implementar con algoritmos como el algoritmo de *Retropropagación del error*.

- **Aprendizaje no supervisado:** la red sólo recibe la entrada. La propia red hace una clasificación basándose en la redundancia entre las muestras. Existen redes como la *máquina de Boltzman* o la *máquina de Cauchy* que siguen este tipo de aprendizaje. Y se puede implementar con algoritmos tales como el *aprendizaje Hebbiano.*

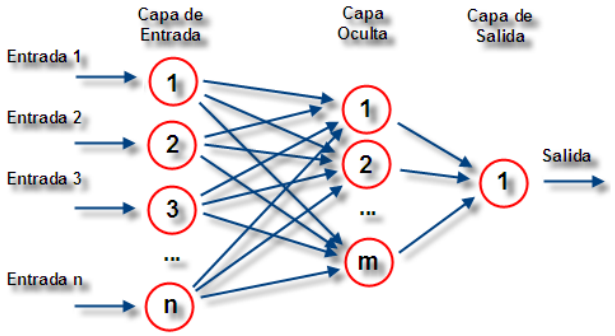
- **Redes híbridas:** llevan a cabo un aprendizaje mixto en el que se usa aprendizaje supervisado para lograr la convergencia. Las *redes de base radial* son de este tipo.

- **Aprendizaje reforzado:** no dispone de un ejemplo completo del comportamiento deseado. Existe un supervisor que indica si la salida proporcionada se acerca a la deseada o no, y en función de ello se ajustan los pesos.

3.2. PERCEPTRÓN MULTICAPA [11] [13] [14] [15]

Existen muchos modelos de red neuronal, pero nos vamos a centrar en el *perceptrón multicapa*, ya que es el modelo de red implementado en este Trabajo fin de grado.

El perceptrón multicapa está formado por una capa de entrada, una capa de salida, y una serie de capas ocultas entre ambas.

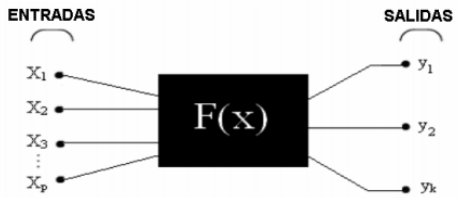


La capa de entrada está formada por las neuronas que introducen los patrones de entrada en la red. En esta capa, no existe nigún tipo de procesamiento por parte de las neuronas.

En las capas ocultas se realiza el procesamiento no lineal de los datos recibidos. El número de capas ocultas puede ser fijado inicialmente.

Las neuronas de la capa de salida son las encargadas de proporcionar la salida de la red para cada entrada.

Se puede entender por lo tanto, la red en su totalidad, como una caja negra, que genera unas ciertas salidas, procedentes de operaciones con las entradas.



Este tipo de red utiliza el **algoritmo de retropropagación** en su entrenamiento. Este tipo de algoritmo consiste en, una vez establecidos los valores de las neuronas de la capa de salida, ir calculando los valores para el resto de neuronas de la red, siguiendo por la capa oculta hasta llegar a la capa de entrada.

Por lo tanto, el proceso de aprendizaje en este tipo de redes queda ordenado de la siguiente manera:

1. Inicialización de pesos y biases de la red de manera aleatoria.
2. Presentando la entrada y las etiquetas, se obtiene una salida.
3. Se evalúa el error cometido por la red entre la salida generada y las etiquetas de entrada.
4. Se modifican los pesos para obtener otra salida.

El objetivo es minimizar el error de entrenamiento. El criterio de parada puede ser un número de iteraciones, tiempo de entrenamiento, error umbral, etc…

Pero no sólo es importante saber si la red es capaz de adaptarse a los patrones de entrenamiento. Sino que también es necesario saber si la red, una vez entrenada, es capaz de adaptarse a unos patrones nuevos, desconocidos y sin etiquetar. Ésa es la parte conocida como parte de test de la red.

Hay que señalar que si el entrenamiento no es suficiente o adecuado, las salidas generadas pueden no ser correctas. Debido a la manera de moverse en busca del mínimo error, puede hacer que la búsqueda caiga en un mínimo local, y que la salida presentada no sea la óptima.

Este tipo de red puede ser aplicado en muchos campos, tales como la segmentación de imágenes, la asociación de patrones o la compresión de datos.

3.3. APLICACIONES DE LAS REDES NEURONALES [16]

La naturaleza de las redes neuronales, las hace adecuadas para resolver eficazmente problemas de clasificación o de emparejamiento, disponiendo de un cojunto de ejemplos de entrada. Estos ejemplos pueden estar etiquetados o no.

Por lo tanto, pueden ser usadas en campos como el reconocimiento y síntesis de voz, diagnóstico médico, clasificación y compresión de la información, robótica y control, procesado de imágenes, etc…

4 IMPLEMENTACIÓN DEL RECONOCEDOR HÍBRIDO

Como ya se ha comentado anteriormente, este trabajo se compone de dos grandes bloques. El primero de ellos se encarga de extraer los parámetros MFCC de las muestras de voz, y el siguiente, se encarga de crear y configurar la red neuronal y el decisor.

4.1. EXTRACCIÓN DE CARACTERÍSTICAS

El primer paso del proceso es el de obtener las muestras de voz.

En este caso, dichas muestras se han obtenido de la base de datos ISOLET Testbed. Esta base de datos fue desarrollada por el ICSI **[17]**.

Esta base de datos contiene 7800 letras del alfabeto inglés pronunciadas de manera aislada. Cada letra ha sido pronunciada y grabada dos veces por cada uno de los 150 hablantes que intervinieron. Lo cual hace un total de, aproximadamente, 85 minutos de habla.Las grabaciones se realizaron en laboratorio con un micrófono cancelador de ruidos. La frecuencia de muestreo es de 16000 Hz. Los hablantes que participaron en las grabaciones tenían el inglés como lengua materna y una edad comprendida entre los 14 y los 72 años. La mitad eran hombres y la otra mitad mujeres.Esta base de datos limpia fue contaminada con diferentes tipos de ruido para poder realizar los experimentos en condiciones menos ideales que las del laboratorio.La base de datos, por lo tanto, está dividida en 5 secciones o “folds” limpias de ruido, y otras 5 secciones ruidosas, cuyo ruido ha sido añadido de manera posterior a la grabación de las muestras.

Una vez que tenemos el conjunto de muestras, el siguiente paso es preparar la herramienta que se va a utilizar para poder llevar a cabo la extracción de parámetros. En este caso, se va a utilizar Matlab como herramienta, y es necesaria instalar la toolbox “Voicebox” que nos permita extraer los parámetros deseados.

Una vez que tenemos preparada la herramienta Matlab, el siguiente paso es configurar la manera de extraer los parámetros.

El tipo de filtrado a llevar es un filtrado triangular tipo Hamming. Para cada frase, se usan 40 bandas ordenadas en frecuencia según la escala MEL.Para cada muestra se extraen los parámetros MFCC, los parámetros MFCC-Delta y los parámetros MFCC-Delta. En total, 39 parámetros para cada muestra.

Esta operación se hace para todas las frases de cada fold. Cuando se termina con la última frase de un fold, se comienza con la primera frase del siguiente fold. De esta manera, se obtiene un fichero con una de las dimensiones 39 y la otra dimensión igual al producto del número de frases (7800) por el número de tramas de cada frase. Este número será variable con cada frase, ya que depende de su duración temporal.

De manera paralela, se van guardando en una matriz el número de frase y el número de trama de cada muestra.

Con estas dos matrices, se forma una sola matriz que es la unión de ambas. En esta nueva matriz, cada fila representa una muestra. Cada fila tendrá por lo tanto 41 componentes. La primera componente de una fila indica el número de frase, la segunda, el número de trama de cada frase, las siguientes 13 componentes serán los coeficientes MFCC, las siguientes 13 componentes serán los MFCC-Delta y por último, las siguientes 13, serán los coeficientes MFCC.

Por lo tanto, al finalizar esta fase, tendremos un fichero .ascii en el que una de las dimensiones será 41, y la otra será la suma de cada frase por el número de tramas usadas en esa frase.

4.2. CREACIÓN Y ENTRENAMIENTO DEL RECONOCEDOR [18]

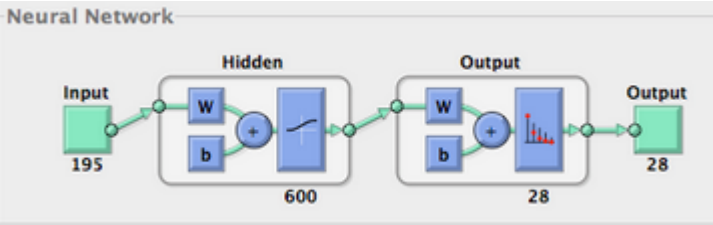
Para poder llevar a cabo esta parte del proceso, se va a hacer uso de la toolbox de Matlab “nnet”, que proporciona las herramientas necesarias para la configuración y desarrollo necesarios.

En este experimento, se usa como reconocedor un perceptrón multicapa. Este perceptrón tendrá una capa de entrada, una capa oculta y una capa de salida.

El número de neuronas de la capa oculta, así como el número de iteraciones son factores clave tanto para el resultado del reconocedor, como para el tiempo que tarda el proceso en ejecutarse. Es necesario conseguir un compromiso entre el resultado y el tiempo de ejecución. En este caso se ha fijado el número de neuronas que forman la capa oculta en 600, y el número de iteraciones en 30 iteraciones. Se han fijado estos valores, ya que de manera experimental, se ha comprobado que a partir de estos valores no se produce una mejora sustancial.

El siguiente parámetro a fijar son las funciones de transferencia que se usan en la capa oculta y en la capa de salida. En la capa oculta se ha utilizado la función “logsig”, que devuelve como salida, valores dentro del intervalo[0,1]. En la capa de salida, la función de transferencia elegida ha sido la función “softmax”. El motivo de elegir esta función ha sido debido a que las salidas de esta función son valores pertenecientes al intervalo [0,1], y que para cada componente, la suma de todos ellos es 1. Por lo tanto, estos valores de salida pueden ser interpretados como probabilidades de pertenencia de cada entrada a cada posible grupo de salida.

A excepción de saber porqué la entrada y la salida tienen esas dimensiones, lo que será explicado en el siguiente apartado, el perceptrón multicapa creado, con la configuración llevada a cabo hasta el momento, queda de la siguiente manera:



El siguiente paso en la configuración de la red neuronal es elegir la forma de inicializar los pesos y biases. Este paso es muy importante, ya que es necesario que en la primera iteración los pesos y biases tengan un valor. Para este cometido se usa la función por defecto “initlay”. Esta función inicializa los pesos y biases de manera aleatoria con valores entre 0 y 1.

El entrenamiento, consta de tres partes. La parte de entrenamiento como tal, la parte de validación y la parte de test. Para cada una de estas partes, se usan conjuntos de datos diferentes. La manera de elegir los datos para formar estos conjuntos a partir del conjunto grande, se define con la función de división de datos. Para este cometido, se ha elegido la función “dividerand”. Esta función elige los datos para formar los tres subconjuntos de manera aleatoria. Lo cual es lo más adecuado, ya que los datos est

Una de las principales funciones a configurar es la función de entrenamiento de la red neuronal. Se ha elegido la función “trainscg” (Scaled conjugate gradient backpropagation). Se usa esta función de entrenamiento debido a que hace uso del algoritmo de backpropagation y a que se basa en el gradiente para moverse dentro de la función de error. También se ha elegido porque está indicada para conjuntos con una gran cantidad de datos, como es nuestro caso.

Por último queda definir la función de error. Ésta es la función que calcula la diferencia entre la salida proporcionada y la etiqueta deseada en cada iteración, para moverse en una dirección u otra y modificar así los pesos. Éste es uno de los parámetros que se han variado para hacer las distintas pruebas. Un conjunto de pruebas se han hecho con la función de error configurada como *entropía cruzada*. Ésta función calcula el error entre la salida proporcionada y las etiquetas deseadas de la siguiente manera:

$$\mathrm{H}(p,q) = -\sum_{x} p(x)\,\log q(x).$$

Después de obtener el error como la entropía cruzada entre las dos distribuciones de probabilidad, aplica una operación sobre el error para decidir en qué dirección moverse. Estas operación puede ser alguna de estas cuatro: *meansqr* (media de los errores al cuadrado), *meanabs* (media de los módulos de los valores), *sumsqr* (suma de los errores al cuadrado) o *sumabs* (suma de los módulos de los errores).

La otra función de error que se ha usado además de *entropía cruzada* ha sido la función conocida como *msereg.* Esta función es una variación de la función de error *mse* (media de los errores al cuadrado). La manera de calcular de *msereg* sigue la siguiente expresión:

$$msereg = \gamma \cdot mse + (1-\gamma)msw$$

$$msw = \frac{1}{K}\sum_{j=1}^{K} w_j^2$$

Por lo tanto, es necesario variar el valor de gamma para darle más importancia a la media de los errores al cuadrado (mse) o al sumatorio de los pesos.

5 EXPERIMENTOS

Para llevar a cabo los experimentos se ha usado siempre el mismo conjunto de entrenamiento. Este conjunto de entrenamiento contiene los coeficiente desde la frase 1561 hasta la frase 7176. Estos parámetros se obtienen del conjunto total obtenido en la fase de extracción de parámetros, en el que estaban los parámetros de las 7800 frases. Para formar el conjunto de entrenamiento, de las frases seleccionadas, nos quedamos sólo con las 39 columnas de los parámetros, deshaciéndonos de las columnas que indican el número de frase y el número de trama dentro de esa frase. Por lo tanto, para cada frase, vamos a tener un conjunto de vectores de 39 componentes cada uno. Por ejemplo, si suponemos que una frase tiene 14 muestras, el conjunto formado por esa frase tendría la siguiente forma:

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14

Se ha llevado a cabo una operación con cada uno de estos conjuntos de cada frase conocido. Se le ha añadido a cada conjunto un contexto. Esta operación consiste en que de cada frase, se le añaden un número de muestras al conjunto de la misma frase, para que la red neuronal, a la hora de ser entrenada, disponga de más muestras de dicha frase. Si seguimos con el ejemplo de la frase anterior, con 14 muestras, y suponemos un contexto de 5, como utilizado en la implementación el conjunto final para esta frase quedaría así:

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V2	V3	V4	V5	V6	V7	V8	V9	V10	V11
V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
V5	V6	V7	V8	V9	V10	V11	V12	V13	V14

Por lo tanto, el número de filas de este conjunto será 39*contexto. En este caso, como hemos utilizado un contexto de 5, el número de filas de este conjunto será de 195. Tras hacer esto con todas las frases, se unen todos los conjuntos formados por contexto, en un único grupo que una los conjuntos de todas las frases desde la frase 1516 hasta la frase 7176.

A este conjunto se le realiza una normalización antes de pasárselo a la red neuronal. La normalización que se lleva a cabo se hace muestra a muestra y consiste en que a cada muestra, se le resta la media y el resultado de esta operación queda dividido entre la varianza. Esta operación se realiza por componentes, no por muestra.

El reconocimiento, se va a hacer por fonema. El conjunto de fonemas usados, con sus respectivas etiquetas, son:

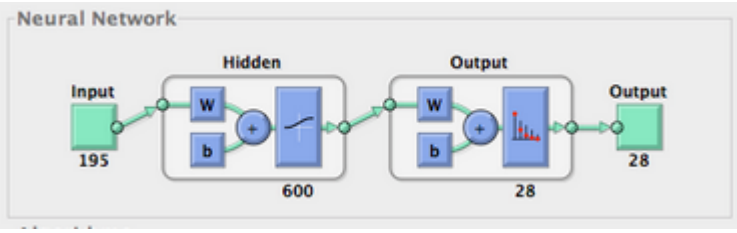
-	aa	ae	ah	ax	ay	b	ch	d	eh	ey	f	iy	jh	k	l	m	n	ow	p	r	s	t	uw	v	w	y	z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

Se dispone de una matriz que nos indica la etiqueta de cada muestra, es decir, de cada trama dentro de cada frase.

La etiqueta para cada muestra viene dada por un vector columna de 28 filas, en el que todas las componentes son 0, excepto una, cuyo valor es 1. Ésta componente con valor 1, nos indica el fonema que representa dicha muestra. Hay que tener en cuenta, que por requisitos de Matlab, la primera coordenada de un vector tiene valor 1 y no 0 como en otras plataformas, por lo que las etiquetas, quedan desplazadas una unidad:

-	aa	ae	ah	ax	ay	b	ch	d	eh	ey	f	iy	jh	k	l	m	n	ow	p	r	s	t	uw	v	w	y	z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28

Por lo tanto, el vector el conjunto de etiquetas tendrá 28 filas y tantas columnas como el vector de componentes. De esta manera, ya podemos entender las dimensiones de la entrada y de la salida de la red neuronal.



Con este conjunto de etiquetas, se realiza la misma operación de contexto que con el conjunto de parámetros.

De esta manera, tenemos dos conjuntos, uno de parámetros y otro de etiquetas para entrenar la red.

Para evaluar el entrenamiento de la red, hay que elegir un conjunto de test. Se han hecho dos conjuntos de pruebas. En las primeras, el conjunto de entrenamiento y el de test era el mismo, y en el segundo conjunto de pruebas, el conjunto de test y el conjunto de entrenamiento eran diferentes.

- **Conjunto de pruebas 1:** El primer objetivo a conseguir con estas pruebas era el de definir una función de error a elegir entre *entropía cruzada* y *msereg*. Por lo tanto, se han hecho pruebas, dejando el resto de parámetros fijos, y cambiando sólo entre *msereg* y variando el valor de gamma, y entre las 4 variantes de la *entropía cruzada.* Los resultados obtenidos son éstos:

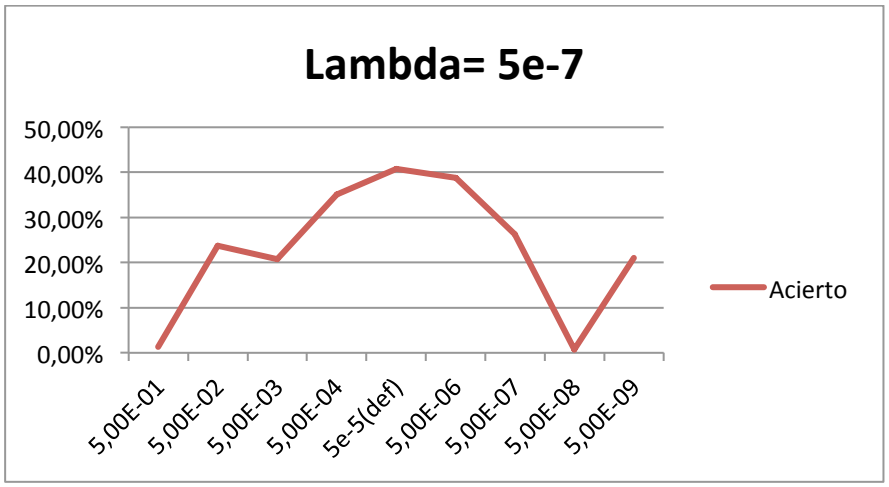
Nº	Conj	Perf Fcn	E0	Ef	G0	Gf	Tiempo	Acierto
1	10	msereg (def)	0,0784	0,0613	0,0075	0,00833	43,59	29,90%
2	10	CrossEnt(meanab	128	128	0,0105	0,0105	42,35	7,76%
3	10	CrossEnt(meansq	7,94E+04	7,94E+04	0,0103	0,0103	11,44	7,76%
4	10	CrossEnt(sumabs	9,33E+05	9,33E+05	0,0104	0,0104	46	7,76%
5	10	CrossEnt(sumsqr)	5,79E+08	5,79E+08	0,0104	0,0104		7,76%
6	10	msereg(0)	0,242	2,37E-28	0,0027	8,40E-17	10,15	1,94%
7	10	msereg(0.1)	0,224	0,00279	0,0026	0,00197	45,06	39,63%
8	10	msereg(0.2)	0,206	0,00618	0,0027	0,0101	46,43	35,90%
9	10	msereg(0.3)	0,187	0,0257	0,0031	0,00549	47,2	24,97%
10	10	msereg(0.4)	0,169	0,0212	0,0037	0,01	46,35	20,22%
11	10	msereg(0.5)	0,151	0,0134	0,0044	0,00164	44,37	40,80%
12	10	msereg(0.6)	0,133	0,017	0,0051	0,0155	42	40,78%
13	10	msereg(0.7)	0,115	0,0695	0,0059	0,00241	46,47	20,41%
14	10	msereg(0.8)	0,0966	0,0819	0,0067	0,00691	46,36	25,32%
15	10	msereg(0.9)	0,0784	0,0642	0,0075	0,00486	45,56	25,26%
16	10	msereg(1)	0,0602	0,0396	0,0084	0,00439	45,49	31,53%

En base a estos resultados se decide usar como función de error la función *msereg* con su parámetros *gamma=0,5.*
De ahora en adelante, todas las pruebas que se hagan en este conjunto de pruebas serán con esta configuración de la función de error.

Una vez fijada la función de error lo que se varía son los dos parámetros que nos permite variar la función de entrenamiento: sigma y lambda. El primero de ellos, sigma, es el que determina el cambio en los pesos; su valor por defecto es de 5e-5. El segundo parámetro, lambda, es el parámetro que se usa para la regulación del Hessiano; su valor por defecto es de 5e-7.
La estrategia a seguir ahora es ir fijando diferentes valores de lambda e ir haciendo un barrido entre los valores de sigma.
Se empieza con el valor de lambda por defecto, 5e-7, y los resultados obtenidos son:

Nº	Conj	Sigma	E0	Ef	G0	Gf	Tiempo	Acierto
17	10	5,00E-01	0,151	0,0341	0,0044	0,00339	41,45	1,27%
18	10	5,00E-02	0,151	0,0237	0,0044	0,00685	46,07	23,74%
19	10	5,00E-03	0,151	0,0348	0,0044	0,00319	45,12	20,73%
20	10	5,00E-04	0,151	0,0153	0,0044	0,00984	45,02	35,06%
21	10	5e-5(def)	0,151	0,0134	0,0044	0,00164	44,37	40,80%
22	10	5,00E-06	0,151	0,0139	0,0044	0,00737	44,49	38,81%
23	10	5,00E-07	0,151	0,132	0,0044	0,00479	43,33	26,29%
24	10	5,00E-08	0,151	0,0178	0,0044	0,00632	46,16	0,69%
25	10	5,00E-09	0,151	0,0623	0,0044	0,00102	48,02	21,03%

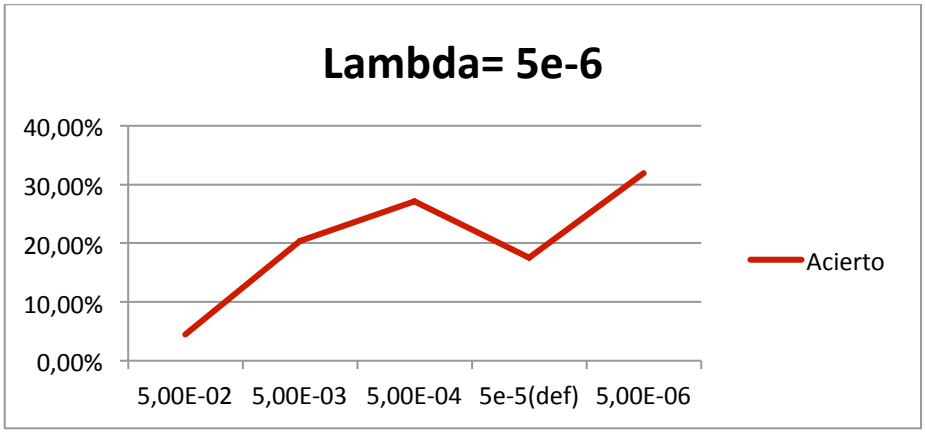
Si representamos el porcentaje de acierto en función del valor de sigma:



Fijamos ahora el valor de lambda a 5e-6. Y de nuevo, hacemos un barrido con los valores de sigma. Los resultados obtenidos son:

Nº	Conj	Sigma	E0	Ef	G0	Gf	Tiempo	Acierto
26	10	5,00E-02	0,151	0,0243	0,0044	0,00943	45,57	4,41%
27	10	5,00E-03	0,151	0,0292	0,0044	0,00011	47,4	20,35%
28	10	5,00E-04	0,151	0,0158	0,0044	0,00628	46,51	27,09%
29	10	5e-5(def)	0,151	0,103	0,0044	0,0046	46,48	17,53%
30	10	5,00E-06	0,151	0,0153	0,0044	0,00749	45,32	31,96%

Representando el porcentaje de acierto en función del valor de sigma, nos queda esta gráfica:

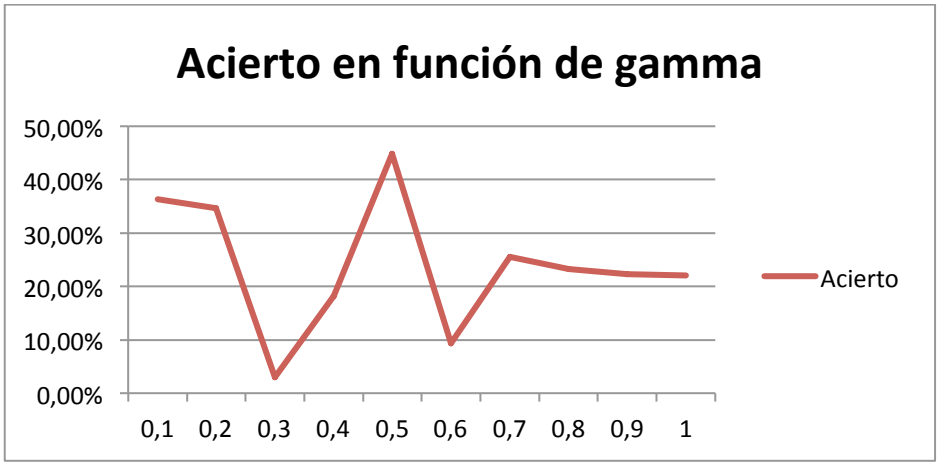


Por lo tanto vemos que la mejor combinación de parámetros para este conjunto de pruebas, es la establecida en la prueba número 21, en la que el porcentaje de acierto es de un 40,1%.

Como estas pruebas no resultan demasiado satisfactorias, se decide cambiar de estrategia, y pasar al segundo conjunto de pruebas.

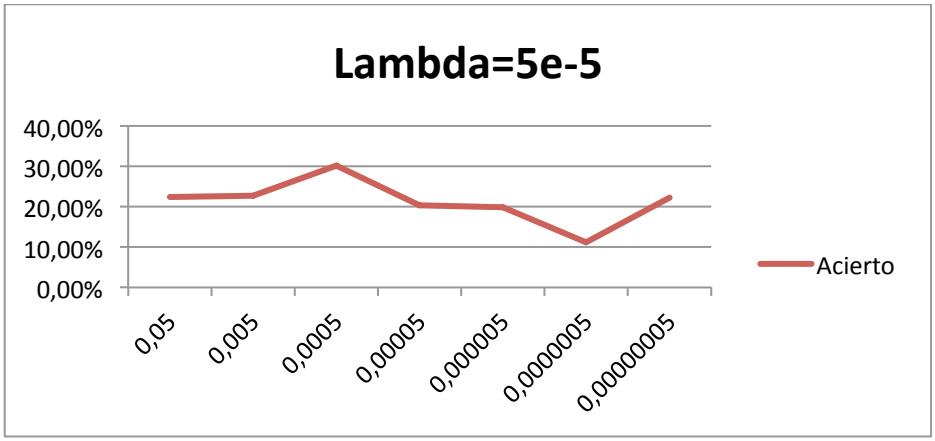
- **Conjunto de pruebas 2**: A diferencia del anterior conjunto de pruebas, en éste, el conjunto de datos utilizados para el entrenamiento es diferente que el conjunto de datos usados para test, para evaluar la red.
  El conjunto de entrenamiento es el mismo en ambos casos, lo que varía es el conjunto de test. Concretamente, el conjunto de test utilizado ahora comprende los parámetros desde la primera frase hasta la frase 1560, con todas las tramas de cada frase.
  Habiendo fijado previamente *msereg* como función de error, se hacen pruebas para determinar el valor de su parámetro, gamma. Los resultados, en función del valor de gamma,en tabla y graficados son:

Nº	Conj Red	Conj Prueba	Gamma	Sigma	Lambda	E0	Ef	G0	Gf	Tiempo	Acierto
1	10	test	0,1	5e-5(def)	5e-7(def)	0,224	0,0028	0,0026	0,0016	39,59	36,35%
2	10	test	0,2	5e-5(def)	5e-7(def)	0,206	0,0056	0,0027	0,0019	46,21	34,69%
3	10	test	0,3	5e-5(def)	5e-7(def)	0,187	0,0188	0,0032	0,004	43,02	2,97%
4	10	test	0,4	5e-5(def)	5e-7(def)	0,169	0,0188	0,0037	0,0049	42,52	18,20%
5	10	test	0,5	5e-5(def)	5e-7(def)	0,151	0,0159	0,0044	0,0041	47,09	44,80%
6	10	test	0,6	5e-5(def)	5e-7(def)	0,133	0,0666	0,0052	0,001	45,34	9,35%
7	10	test	0,7	5e-5(def)	5e-7(def)	0,115	0,0992	0,0059	0,0052	43,34	25,49%
8	10	test	0,8	5e-5(def)	5e-7(def)	0,097	0,0802	0,0067	0,004	46,28	23,20%
9	10	test	0,9	5e-5(def)	5e-7(def)	0,078	0,062	0,0075	0,0075	44,35	22,31%
10	10	test	1	5e-5(def)	5e-7(def)	0,06	0,0412	0,0084	0,0095	46,45	22,09%

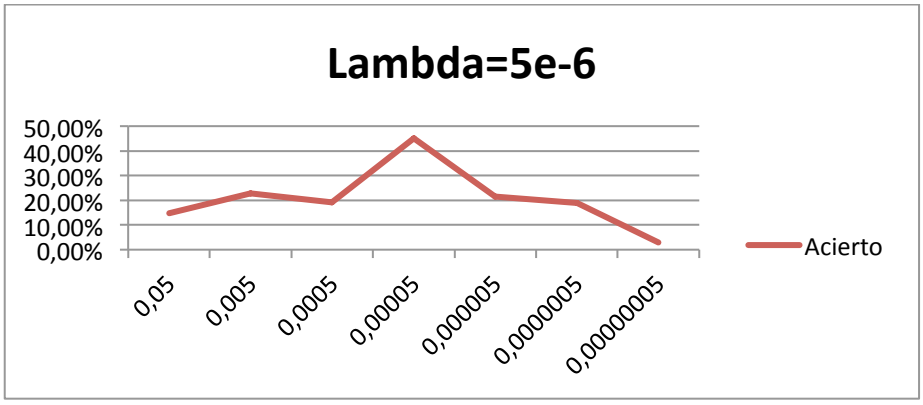


A la vista de los resultados, se fija el valor de gamma a 0,5. Nuevamente, se hace un barrido de los valores de sigma dentro de cada posible valor de lambda. Los resultados obtenidos, en función del valor de sigma, son:

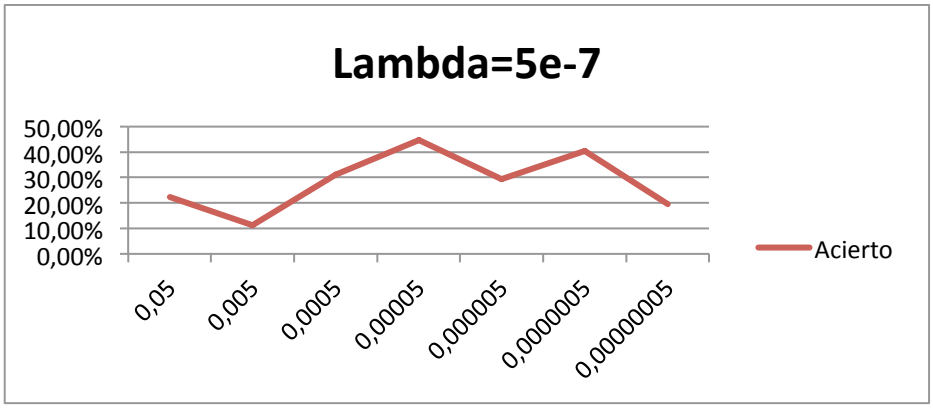
Nº	Conj Red	Conj Prueba	Gamma	Sigma	Lambda	E0	Ef	G0	Gf	Tiempo	Acierto
8	10	test	0.5	0,05	0,00005	0,151	0,0307	0,0044	0,0023	46,16	22,33%
9	10	test	0.5	0,005	0,00005	0,151	0,0293	0,0044	0,0049	45,3	22,76%
10	10	test	0.5	0,0005	0,00005	0,151	0,0163	0,0044	0,0132	45,19	30,17%
11	10	test	0.5	0,00005	0,00005	0,151	0,0288	0,0044	0,0006	48,5	20,39%
12	10	test	0.5	0,000005	0,00005	0,151	0,0222	0,0044	0,0026	44,2	19,94%
13	10	test	0.5	5E-07	0,00005	0,151	0,0214	0,0044	0,0117	45,18	11,18%
14	10	test	0.5	5E-08	0,00005	0,151	0,0284	0,0044	0,0087	44,42	22,18%



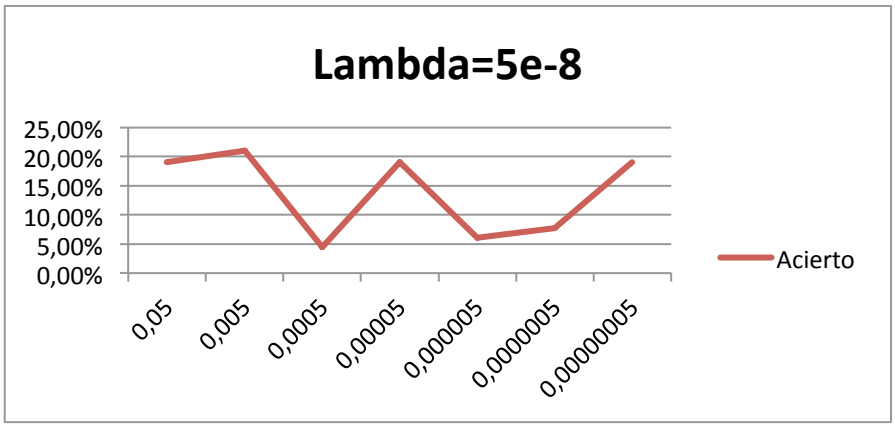
Nº	Conj Red	Conj Prueba	Gamma	Sigma	Lambda	E0	Ef	G0	Gf	Tiempo	Acierto
1	10	test	0.5	0,05	0,000005	0.151	0.0287	0.00436	0.0140	49,57	14,65%
2	10	test	0.5	0,005	0,000005	0.151	0.0165	0.00440	0.00617	50,38	22,81%
3	10	test	0.5	0,0005	0,000005	0.151	0.0161	0.00441	0.00418	48,55	19,05%
4	10	test	0.5	0,00005	0,000005	0,151	0,014	0,0044	0,0034	47,44	45,07%
5	10	test	0.5	0,000005	0,000005	0,151	0,0153	0,0044	0,0018	49,17	21,49%
6	10	test	0.5	5E-07	0,000005	0,151	0,0267	0,0044	0,0016	48,43	18,95%
7	10	test	0.5	5E-08	0,000005	0,151	0,0269	0,0044	0,006	49,29	2,92%



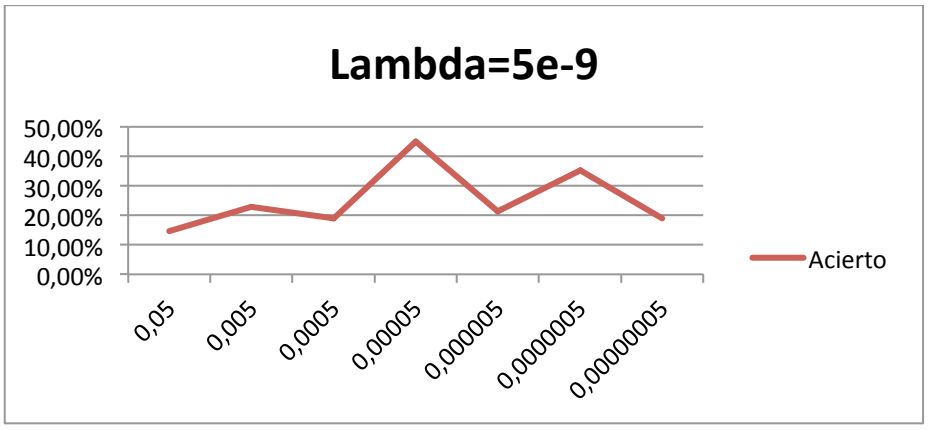
Nº	Conj Red	Conj Prueba	Gamma	Sigma	Lambda	E0	Ef	G0	Gf	Tiempo	Acierto
11	10	test	0.5	0,05	5e-7(def)	0,151	0,052	0,0044	0,0025	47,54	22,31%
12	10	test	0.5	0,005	5e-7(def)	0,151	0,106	0,0044	0,0032	43,49	11,25%
13	10	test	0.5	0,0005	5e-7(def)	0,151	0,0159	0,0044	0,0018	45,09	31,07%
14	10	test	0,5	5e-5(def)	5e-7(def)	0,151	0,0159	0,0044	0,0041	47,09	44,80%
15	10	test	0.5	0,000005	5e-7(def)	0,151	0,0156	0,0044	0,02	41,58	29,36%
16	10	test	0.5	5E-07	5e-7(def)	0,151	0,013	0,0044	0,0068	46,44	40,40%
17	10	test	0.5	5E-08	5e-7(def)	0,151	0,137	0,0044	0,003	44,04	19,59%



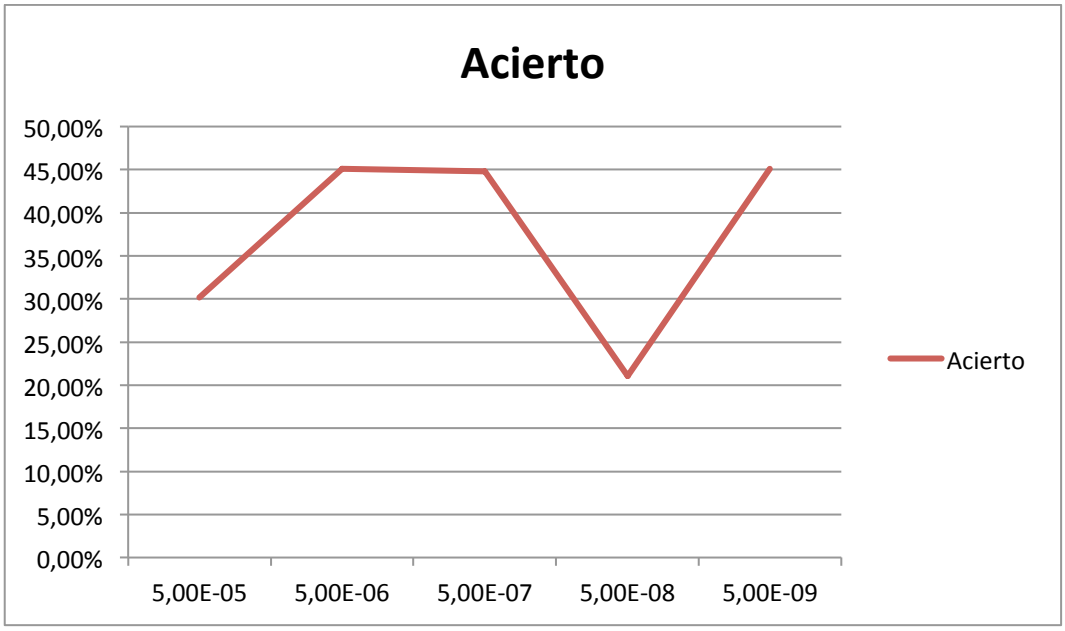
Nº	Conj Red	Conj Prueba	Gamma	Sigma	Lambda	E0	Ef	G0	Gf	Tiempo	Acierto
18	10	test	0.5	0,05	0,00000005	0,151	0,0273	0,0044	0,0012	47,06	19,04%
19	10	test	0.5	0,005	0,00000005	0,151	0,0167	0,0044	0,0102	47,47	21,06%
20	10	test	0.5	0,0005	0,00000005	0,151	0,0265	0,0044	0,0115	46,08	4,46%
21	10	test	0,5	5e-5(def)	0,00000005	0,151	0,0028	0,0044	0,0008	45,56	19,02%
22	10	test	0.5	0,000005	0,00000005	0,151	0,0179	0,0044	0,0076	45,57	6,12%
23	10	test	0.5	5E-07	0,00000005	0,151	0,0328	0,0044	0,0031	45,58	7,74%
24	10	test	0.5	5E-08	0,00000005	0,151	0,0165	0,0044	0,0036	46,36	19,05%



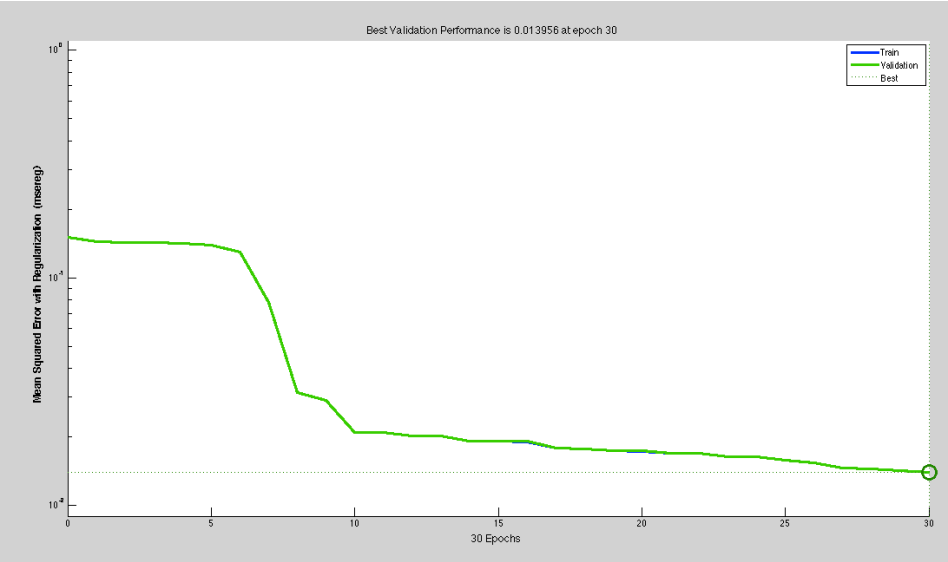
Nº	Conj Red	Conj Prueba	Gamma	Sigma	Lambda	E0	Ef	G0	Gf	Tiempo	Acierto
25	10	test	0.5	0,05	0,000000005	0,151	0,0287	0,0044	0,014	44,55	14,65%
26	10	test	0.5	0,005	0,000000005	0,151	0,0165	0,0044	0,0062	46,38	22,81%
27	10	test	0.5	0,0005	0,000000005	0,151	0,0161	0,0044	0,0042	44,59	19,05%
28	10	test	0,5	5e-5(def)	0,000000005	0,151	0,014	0,0044	0,0034	44,1	45,07%
29	10	test	0.5	0,000005	0,000000005	0,151	0,0431	0,0044	0,0004	45,46	21,34%
30	10	test	0.5	5E-07	0,000000005	0,151	0,0152	0,0044	0,0059	43,57	35,18%
31	10	test	0.5	5E-08	0,000000005	0,151	0,0313	0,0044	0,0002	45,44	19,05%



Cogiendo el mejor valor como representante para cada valor de lambda, podemos representar el acierto en función de lambda.

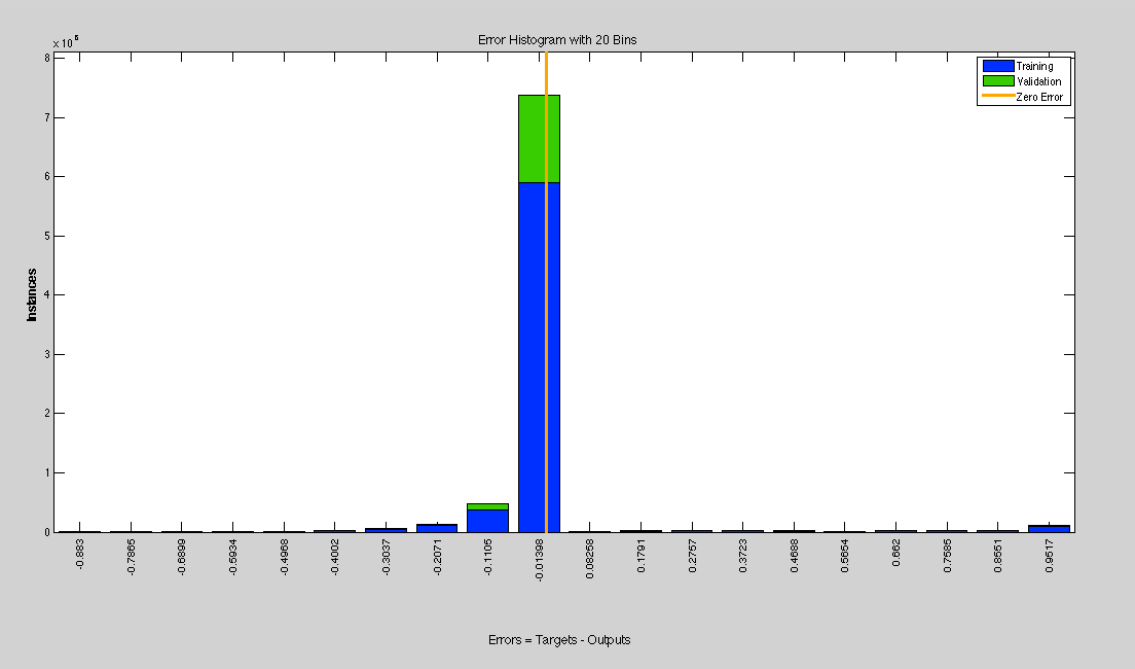


Tras todas estas pruebas, el mejor resultado obtenido ha sido un porcentaje de acierto de 45,1%. Este resultado se ha obtenido con la prueba número 28 con gamma=0,5, sigma=5e-5 y lambda=5e-9.
Analizamos ahora las gráficas proporcionadas por Matlab para esta prueba concreta. Primero analizamos la evolución del error:



En la gráfica anterior de la evolución del error vemos que la mayor parte de la mejora se produce en las 10 primeras iteraciones, donde disminuye el error de manera bastante brusca. A partir de ahí, el error sigue disminuyendo, aunque de una manera más lenta.

Vemos ahora el histograma de los errores entre las salidas y las etiquetas deseadas:



En este histograma podemos ver que la mayoría de diferencias son muy pequeñas, casi nulas. Son las diferencias grandes, las que están más alejadas del cero, las que hacen que el acierto no sea mayor, aunque la densida de estas muestras no sea tan alta.

6 EXPERIMENTOS

6.1. CONCLUSIONES

Analizando la colección de pruebas realizadas, se ve que los resultados obtenidos en ambas, son muy parecidos.

El conjunto de pruebas que más se asemeja a una prueba real, es el segundo conjunto de pruebas. En este conjunto, los porcentajes de acierto son superiores a los porcentajes de acierto del primer conjunto de pruebas. Éste es un buen resultado, ya que como se ha dicho antes, se asemeja más a una prueba real en la que el conjunto de entrenamiento es diferente al conjunto de test.

Habiendo fijado la función de error, se podría hacer un barrido más fino de los parámetros gamma, sigma y lambda, para obtener mejores resultados. Se podría también, modificar algunas funciones como la función que inicializa los pesos, o la función que genera los conjuntos de entrenamiento, de validación y de test, dentro de la etapa de entrenamiento.

Incluso, se podrían hacer el mismo tipo de pruebas, con otra toolbox diferente a la usada en este trabajo.

6.2. DECODIFICACIÓN DE VITERBI

Una de las posibles líneas futuras en las que enfocar este trabajo, es el reconocimiento de frases, en lugar de reconocimiento de fonemas.

Otra posible línea de trabajo se basa en la introducción del algoritmo de Viterbi para llevar a cabo la decodificación, lo que se puede entender como decisor. Este algoritmo es un algoritmo secuencial, que permite encontrar las secuencias de estado más probables de un conjunto estados. Esto lo consigue asignando una etiqueta a cada rama entre dos estados y se va formando el camino de menor distancia. La decisión se toma siempre al final de la secuencia.

Este tipo de decisor puede ser muy útil en el reconocimiento de fonemas, ya que hay ciertas transiciones que se repiten con mayor frecuencia que otras.

6.3. EXPERIENCIA CON TOOLBOX NNET

Para la selección de la toolbox que nos permita implementar el perceptrón multicapa sobre Matlab, se han tenido en cuenta algunos criterios, ordenados por orden de importancia:

- Las salidas de la red deben de ser de tipo softmax.
- El entrenamiento debe poder realizarse según el criterio de error de entropía cruzada.
- Se debe poder iniciar con pesos predefinidos.
- Se debe poder utilizar un conjunto de validación cruzada para parar el entrenamiento.

Encontramos varias toolboxes que cumplían la mayoría de los requisitos **[3] [4] [5]**. Se optó por elegir la toolbox nativa de Matlab “nnet” para dicho cometido.

Se ha elegido esta toolbox, debido a que, al ser la proporcionada por Matlab asegura una total compatibilidad y una cierta estabilidad.

Sin embargo, la experiencia con el trato de esta toolbox no ha sido del todo amigable. El primer problema que nos encontramos era que dicha toolbox no cumple el segundo requisito, lo cual ha provocado que los resultados no fueran los esperados. Además, durante el desarrollo de este trabajo han surgido diferentes problemas en la ejecución, cuyo origen ha sido muy complicado de localizar. Esto problemas han ralentizado la marcha del desarrollo de este problema.

7 PRESUPUESTO

El presupuesto de este proyecto se ha elaborado siguiendo la plantilla proporcionada por la universidad para dicho cometido **[19]**.

El presupuesto se compone de cuatro fuentes de costes directos: costes de personal, costes de equipos, costes de subcontratación de tareas y otros costes directos del proyecto.

Además, se tiene en cuenta que la tasa de costes indirectos es de un 20%.

- **Costes de personal**: para este proyecto ha sido necesario el trabajo de dos Ingeniero Senior, la Dra. Ascensión Gallardo Antolín y la Dra. Carmen Peláez Moreno, en el puesto de Tutoras del proyecto. También ha sido necesario el trabajo de un Ingeniero, Miguel Ángel Mota Martín, en el puesto de desarrollador principal.
  Estas tres personas han trabajado durante 5 meses, siendo el coste mensual de los Ingenieros Senior de 4.289,54€ y el coste mensual del Ingeniero 2.649,39€. El total de costes de personal durante los 5 meses de trabajo asciende a 56.367,35€.

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Gallardo Antolín, Ascensión		Ingeniero Senior	5		21.447,70	
Peláez Moreno, Carmen		Ingeniero Senior	5	4.289,54	21.447,70	
Mota Martín, Miguel Ángel		Ingeniero	5	2.694,39	13.471,95	
					0,00	
					0,00	
		Hombres mes 15		Total	56.367,35	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

- **Costes de equipos:** para el desarrollo de este proyecto ha sido necesaria la compra de un nuevo equipo para poder la ejecución del proyecto. El coste de este equipo sin IVA ha sido de 1.185€, y su uso ha sido dedicado de manera exclusiva para este trabajo.

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Ordenador	1.185,00	100	5	60	98,75
				60	0,00
				60	0,00
				60	0,00
				60	0,00
					0,00
				Total	98,75

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

- **Costes de subcontratación de tareas:** para este proyecto no ha sido necesario hacer uso de ninguna subcontratación de tareas.

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
N/A		
	Total	0,00

- **Otros costes directos del proyecto:** en este caso, no ha habido ningún tipo de coste extra directo en el proyecto.

OTROS COSTES DIRECTOS DEL PROYECTO[e)]

Descripción	Empresa	Costes imputable
N/A		
	Total	0,00

[e)] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas,

Por lo tanto, teniendo en cuenta los costes directos, y asumiendo una tasa de costes indirectos del 20%, el presupuesto total de este proyecto hace un total de 67.759€.

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	56.367
Amortización	99
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	11.293
Total	67.759

8 REFERENCIAS

Para el desarrollo de la herramienta software se ha hecho consulta de manera frecuente a dos herramientas:

1. **MathWorks**
   http://www.mathworks.es/index.html

2. **MatlabCentral**
   http://www.mathworks.es/matlabcentral/

Para la selección de toolbox para la implementación de la ANN se han consultado la documentación de diferentes toolboxes:

3. **Netlab. Aston University**
   http://www1.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/overview-and-examples/

4. **MathWorks. Neural Network Toolbox**
   http://www.mathworks.es/products/neural-network/description1.html

5. **DTU Toolbox**
   http://bsp.teithe.gr/members/downloads/DTUToolbox.html

6. **Wikipedia. MFCC**
   http://es.wikipedia.org/wiki/MFCC

7. **Memoria de proyecto. Extracción de características**
   http://bibing.us.es/proyectos/abreproy/12054/fichero/MEMORIA%252F8.Cap%EDtulo+3.pdf

8. **Wikipedia. Modelos ocultos de Markov**
   http://es.wikipedia.org/wiki/Modelo_oculto_de_M%C3%A1rkov

9. **Neural networks: a comprehensive foundation**
Haykin, Simon S.
2nd edition

10. **UC3M. J. M. Marín. Introducción a las redes neuronales aplicadas**
http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema3dm.pdf

11. **Wikipedia. Red neuronal artificial**
http://es.wikipedia.org/wiki/Red_neuronal_artificial

12. **IAC. Introducción a las Redes Neuronales**
http://www.iac.es/sieinvens/SINFIN/Sie_Courses_PDFs/NNets/confiac.pdf

13. **Wikipedia. Perceptrón Multicapa**
http://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa

14. **UDLAP. Perceptrón Multicapa**
http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/mejia_s_ja/capitulo3.pdf

15. **UC3M. Perceptrón Multicapa**
http://eva.evannai.inf.uc3m.es/et/docencia/rn-inf/documentacion/Tema3-MLP.pdf

16. **Universidad Tecnológica Nacional. Redes Neuronales: conceptos básicos y aplicaciones**
ftp://decsai.ugr.es/pub/usuarios/castro/Material-Redes-Neuronales/Libros/matich-redesneuronales.pdf

17. **Escuela Superior de Ingeniería de Bilbao. Redes Neuronales Artificiales y sus Aplicaciones**
http://cvb.ehu.es/open_course_ware/castellano/tecnicas/redes_neuro/contenidos/pdf/libro-del-curso.pdf

18. **ICSI. Isolet**
http://www1.icsi.berkeley.edu/Speech/papers/gelbart-ms/hybrid-testbed/

19. Memoria de proyecto
http://earchivo.uc3m.es/bitstream/10016/8488/1/Proyecto%20Redes%20neuronales%20GUI.pdf

20. UC3M. Plantilla para la elaboración del presupuesto de un proyecto
http://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera